



# **PARAMETER TOOLS**

## Supplement To Patch Description

Patch XT\*7.3\*26

August 2001

Documentation Revised: July 2004



# Revision History

## Documentation Revisions

The following table displays the revision history for this document. Revisions to the documentation are based on patches and new versions released to the field.

Date	Revision	Description	Author
08/01	1.0	Initial Patch XT*7.3*26 Supplemental Documentation creation.	Wally Fort, Oakland, CA OIFO; Marcia Insley, Salt Lake City, UT OIFO
12/12/03	2.0	Reformatted Patch XT*7.3*26 Supplemental Documentation and updated API content.	Wally Fort, Oakland, CA OIFO and Thom Blom, Oakland, CA OIFO
07/01/04	2.1	Updated documentation based on changes from Patches XT*7.3*79 and XT*7.3*82.	Wally Fort, Oakland, CA OIFO and Thom Blom, Oakland, CA OIFO, Susan Strack, Oakland, CA OIFO

**Table i: Documentation revision history**

## Patch Revisions

For a complete list of patches related to this software, please refer to the Patch Module on FORUM.

## Revision History

# Contents

Revision History .....	iii
Orientation .....	ix
<b>Chapter 1: User Manual Information.....</b>	<b>1-1</b>
Introduction .....	1-1
Background .....	1-2
Description .....	1-3
Definitions .....	1-3
Entity .....	1-3
Parameter .....	1-4
Instance .....	1-4
Value .....	1-5
Parameter Template .....	1-5
Why Would You Use Parameter Tools? .....	1-5
Example .....	1-6
<b>Chapter 2: Programmer Manual Information .....</b>	<b>2-1</b>
Application Program Interfaces (APIs)—^XPAR Routine.....	2-1
\$\$GET^XPAR—Return An Instance of a Parameter .....	2-1
ADD^XPAR—Add Parameter Value .....	2-3
CHG^XPAR—Change Parameter Value .....	2-5
DEL^XPAR—Delete Parameter Value .....	2-7
EN^XPAR—Add, Change, Delete Parameters .....	2-9
ENVAL^XPAR—Return All Instances of a Parameter .....	2-11
GETLST^XPAR—Return All Instances of a Parameter .....	2-13
GETWP^XPAR—Return Word-processing Text.....	2-15
NDEL^XPAR—Delete All Instances of a Parameter .....	2-17
PUT^XPAR—Add/Update Parameter Instance.....	2-19
REP^XPAR—Replace Instance Value .....	2-21
Application Program Interfaces (APIs)—^XPAREDIT Routine.....	2-24
BLDLST^XPAREDIT—Return All Entities of a Parameter .....	2-24
EDIT^XPAREDIT—Edit Instance and Value of a Parameter .....	2-25
EDITPAR^XPAREDIT—Edit Single Parameter .....	2-26

## Contents

EN^XPAREDIT—Parameter Edit Prompt.....	2-26
GETENT^XPAREDIT—Prompt for Entity Based on Parameter .....	2-27
GETPAR^XPAREDIT—Select Parameter Definition File.....	2-28
TED^XPAREDIT—Edit Template Parameters (No Dash Dividers) .....	2-29
TEDH^XPAREDIT—Edit Template Parameters (With Dash Dividers) .....	2-30
Index .....	Index-1

# Figures and Tables

Table i: Documentation revision history.....	iii
Table ii: Documentation symbol descriptions .....	ix
Table 1-1: Parameter Entities.....	1-3
Table 1-2: Templates—Parameter Tools .....	1-5
Figure 1-1: Adding a parameter template (sample) .....	1-8

## Figures and Tables

# Orientation

## How to Use this Manual

Throughout this manual, advice and instructions are offered regarding the use of Kernel Toolkit and Patch XT\*7.3\*26 software and the functionality it provides for Veterans Health Information Systems and Technology Architecture (VistA) software products.

This manual uses several methods to highlight different aspects of the material:

- Various symbols are used throughout the documentation to alert the reader to special information. The following table gives a description of each of these symbols:

Symbol	Description
	Used to inform the reader of general information including references to additional reading material.
	Used to caution the reader to take special notice of critical information.

Table ii: Documentation symbol descriptions

- Descriptive text is presented in a proportional font (as represented by this font).
- HL7 messages, "snapshots" of computer online displays (i.e., roll-and-scroll screen captures/dialogues) and computer source code, if any, are shown in a *non*-proportional font and enclosed within a box.
  - User's responses to online prompts will be boldface type. The following example is a screen capture of computer dialogue, and indicates that the user should enter two question marks:

Select Primary Menu option: ??

- The "<Enter>" found within these snapshots indicate that the user should press the Enter key on their keyboard. Other special keys are represented within <> angle brackets. For example, pressing the PF1 key can be represented as pressing <PF1>.
- Author's comments, if any, are displayed in italics or as "callout" boxes.



Callout boxes refer to labels or descriptions usually enclosed within a box, which point to specific areas of a displayed image.

- All uppercase is reserved for the representation of M code, variable names, or the formal name of options, field and file names, and security keys (e.g., the XUPROGMODE key).

## How to Obtain Technical Information Online

Exported file, routine, and global documentation can be generated through the use of Kernel, MailMan, and VA FileMan utilities.

-  Methods of obtaining specific technical information online will be indicated where applicable under the appropriate topic. Please refer to the *Kernel Toolkit Technical Manual* for further information.

### Help at Prompts

VistA software provides online help and commonly used system default prompts. Users are encouraged to enter question marks at any response prompt. At the end of the help display, you are immediately returned to the point from which you started. This is an easy way to learn about any aspect of VistA software.

To retrieve online documentation in the form of Help in any VistA character-based product:

- Enter a single question mark ("?") at a field/prompt to obtain a brief description. If a field is a pointer, entering one question mark ("?") displays the HELP PROMPT field contents and a list of choices, if the list is short. If the list is long, the user will be asked if the entire list should be displayed. A YES response will invoke the display. The display can be given a starting point by prefacing the starting point with an up-arrow ("^") as a response. For example, ^M would start an alphabetic listing at the letter M instead of the letter A while ^127 would start any listing at the 127th entry.
- Enter two question marks ("??") at a field/prompt for a more detailed description. Also, if a field is a pointer, entering two question marks displays the HELP PROMPT field contents and the list of choices.
- Enter three question marks ("??") at a field/prompt to invoke any additional Help text stored in Help Frames.

### Obtaining Data Dictionary Listings

Technical information about files and the fields in files is stored in data dictionaries. You can use the List File Attributes option on the Data Dictionary Utilities submenu in VA FileMan to print formatted data dictionaries.

-  For details about obtaining data dictionaries and about the formats available, please refer to the "List File Attributes" chapter in the "File Management" section of the *VA FileMan Advanced User Manual*.

## Assumptions About the Reader

This manual is written with the assumption that the reader is familiar with the following:

- VistA computing environment
- VA FileMan data structures and terminology
- Microsoft Windows
- M programming language

It provides an overall explanation of Kernel Toolkit Patch XT\*7.3\*26. However, no attempt is made to explain how the overall VistA programming system is integrated and maintained. Such methods and procedures are documented elsewhere. We suggest you look at the various VA home pages on the World Wide Web (WWW) for a general orientation to VistA. For example, go to the Veterans Health Administration (VHA) Office of Information (OI) Health Systems Design & Development (HSD&D) Home Page at the following Web address:

<http://vista.med.va.gov/>

## Reference Materials

Readers who wish to learn more about the Kernel Toolkit software should consult the following:

- *Kernel Toolkit Release Notes*
- *Kernel Toolkit Installation Guide*
- *Kernel Toolkit User Manual*
- *Kernel Toolkit Technical Manual*
- The Kernel Toolkit Home Page at the following Web address:  
<http://vaww.vista.med.va.gov/toolkit/index.asp>

This site contains additional information and documentation.

VistA documentation is made available online in Microsoft Word format and Adobe Acrobat Portable Document Format (PDF). The PDF documents *must* be read using the Adobe Acrobat Reader (i.e., ACROREAD.EXE), which is freely distributed by Adobe Systems Incorporated at the following Web address:

<http://www.adobe.com/>

VistA documentation can be downloaded from the Enterprise VistA Support (EVS) anonymous directories or from the Health Systems Design and Development (HSD&D) VistA Documentation Library (VDL) Web site:

<http://www.va.gov/vdl/>



For more information on the use of the Adobe Acrobat Reader, please refer to the *Adobe Acrobat Quick Guide* at the following Web address:

<http://vista.med.va.gov/iss/acrobat/index.asp>



**DISCLAIMER:** The appearance of any external hyperlink references in this manual does not constitute endorsement by the Department of Veterans Affairs (VA) of this Web site or the information, products, or services contained therein. The VA does not exercise any editorial control over the information you may find at these locations. Such links are provided and are consistent with the stated purpose of this VA Intranet Service.

# Chapter 1: User Manual Information

This is the User Manual section of this supplemental documentation for the Parameter Tools software (i.e., Kernel Toolkit Patch XT\*7.3\*26). It will be incorporated into the *Kernel Toolkit User Manual* at a later date.

The intended audience for this chapter is the Information Resource Management (IRM) at a local site. However, it can also be helpful to application developers of Veterans Health Information Systems and Technology Architecture (VistA) software and others in VHA Office of Information (OI) Health Systems Design & Development (HSD&D), Enterprise VistA Support (EVS), and Application Structure and Integration Services (ASIS).

## Introduction

This supplemental documentation is intended for use in conjunction with the Parameter Tools patch (XT\*7.3\*26). This documentation explains the functions available with the use of the Parameter Tools and describes the APIs that are part of the patch. It combines information from the patch description and two Integration Agreements (IAs): 2263 and 2336, as well as providing additional explanatory material and a generic example to illustrate the use of the Parameter Tools.

In brief, the Parameter Tools patch provides a method of managing the definition, assignment, and retrieval of parameters for VistA software applications.

VistA software applications are designed to be used in a variety of ways. Many aspects of hospital activity vary from one hospital to another and thus there are many possible ways software applications can be used that also vary from one institution to another. Each site has its own requirements—its own settings for each software application. IRM staff must modify the software parameters to fit their requirements.

Previously, each software application had its own files and options but no two software applications had the site parameters set up the same way or found in the same place. Thus, when a new software application was released, each site would have to look for the location where the settings were stored for that software. Next, they would have to look to see what settings were available and how to set them. Very little about the parameters was uniform from software to software.

With the Computerized Patient Record System (CPRS) software, the idea was born that a parameter file could be created to export with the software. The CPRS parameter file and parameter utility were subsequently modified to create a generic method of exporting and installing other VistA software applications. Most developers were willing to abandon previous methods and use this tool for software they were developing.

Parameter Tools was designed as a method of managing the definition, assignment, and retrieval of parameters for VistA software. A parameter may be defined for various levels at which you want to allow the parameter described (e.g., software level, system level, division level, location level, user level).

## Background

Whenever you have an entity with many attributes that apply to it, you can do either of the following:

1. Make one big relation to represent that entity.
2. Create a "binary" relation to represent the entity. In the latter case, the relation consists of two columns (thus the term binary), one representing the attribute and the other representing the value for that attribute. So each tuple (i.e., a data type/data object containing two or more components) of the relation represents a single attribute and its associated value.



This works only when the individual attributes are independent observations (have no dependencies on anything other than the key that identifies the entity). Such a relation tends to look a lot like a Windows INI file.

Most of the VistA parameter files were very long lists of independent values that pertained to a single entity. In most cases, this entity was the site or system on which the software was running [similar to an INI file]. In other cases, however, the parameter files had multiples that made things more complex. These multiples generally allow parameters to be defined at levels more specific than the site (e.g., by divisions or hospital location). It seems best to accommodate this by using both an entity identifier and parameter together to name any given value. This yields a relation with a compound key:

Entity | Parameter = Value

Finally, it seems that multiple-valued parameters (e.g., collection times) occur often enough that it is worthwhile to add a field to identify the parameter instance. So the relation becomes:

Entity | Parameter | Instance = Value

This is the relation that the PARAMETERS file (#8989.5) is intended to represent.

Software parameter files frequently maintain parameters that apply to the site, a division, or a location. In addition, many parameters that apply to individual users are kept in the NEW PERSON file (#200). Also, many parameter values are hard-coded in individual software routines for the case when the site has not set up a value for a given parameter. Entity, then, is implemented as a variable pointer.

A given parameter may occur for a variety of entities. In fact, we frequently need to obtain the value of a parameter by following an entity "chain." For example, the Add Orders menu a CPRS user sees may be defined at various levels. Initially, a site generally creates a custom Add Orders menu. Later, hospital locations may each build a custom menu that more specifically meets their needs. Individual users may also have their own Add Orders menus. If no site configuration has been done, the Add Orders menu exported with OE/RR is used. So, when OE/RR needs to display an Add Orders menu, a chain is followed that looks first to see if the user has their own menu. Next, the current location is checked, followed by the site. Finally, if no values exist, the software default menu is used.

In the PARAMETER DEFINITION file (#8989.51), a multiple lists which entities are valid with a given parameter. These entities are also assigned a precedence, so that it is possible to write functions that will "chain" through entities until a value is found, using the proper sequence.

## Description

Patch XT\*7.3\*26 contains a developer toolset that allows creation of software parameters in a central location. Integration Agreements (IAs) 2263 and 2336 define the supported entry points for this application. Kernel Patch XU\*8.0\*201 allows KIDS to transport the parameters.

Parameter Tools is a generic method of handling parameter definition, assignment, and retrieval. A parameter can be defined for various entities where an entity is the level at which you want to allow the parameter defined (e.g., software level, system level, division level, location level, user level, etc.). A developer can then determine in which order the values assigned to given entities are interpreted.

## Definitions

The following are some basic definitions used by Parameter Tools:

### Entity

An entity is a level at which you can define a parameter. The entities allowed are stored in the PARAMETER ENTITY file (#8989.518). Kernel Toolkit patches maintain entries in this file.

The list of allowable entries is as follows:

Prefix	Message	Points To File
PKG	Package	PACKAGE (#9.4)
SYS	System	DOMAIN (#4.2)
DIV	Division	INSTITUTION (#4)
SRV	Service	SERVICE/SECTION (#49)
LOC	Location	HOSPITAL LOCATION (#44)
TEA	Team	TEAM (#404.51)
CLS	Class	USR CLASS (#8930)
USR	User	NEW PERSON (#200)
BED	Room-Bed	ROOM-BED (#405.4)
OTL	Team (OE/RR)	OE/RR LIST (#101.21)

**Table 1-1: Parameter Entities**

Package (PKG), as an entity, allows the software defaults to be handled the same way as other parameters rather than hard-coded.

System (SYS), Division (DIV), Location (LOC), and User (USR) are frequent entries in existing software parameter files (or additions to the NEW PERSON file [#200]).

Service (SRV), Team (TEA), and Class (CLS) are referenced frequently by parameters that pertain to Notifications.

The process of exporting software using this kind of parameters file involves sending:

1. Parameter definitions that belong to the software (entries in the PARAMETER DEFINITION file [#8989.51]).
2. Actual parameter instances that point to the software (entries in the PARAMETERS file [#8989.5] that have an entity that matches the software).

All the other entries in the PARAMETERS file (#8989.5 (those that correspond to entities other than package [PKG]) would never be exported, as they are only valid for the system on which they reside.

## Parameter

A parameter is the actual name under which values are stored. The name of the parameter must be namespaced and it must be unique and start with two uppercase characters. Parameters can be defined to store the typical software parameter data (e.g., the default add order screen in OE/RR), but they can also be used to store graphical user interface (GUI) application screen settings a user has selected (e.g., font or window width). With each parameter, a more readable display name can also be defined. When a parameter is defined, the entities that may set that parameter are also defined. The definition of parameters is stored in the PARAMETER DEFINITION file (#8989.51).

## Instance

An instance is a unique value assigned to an entity/parameter combination. For most parameters, there will only be one instance, that is, instance does not apply and is simply set to "1".

However, a parameter can be multi-valued—it can have more than one instance. More than one value can be assigned to the parameter as it relates to a specific entity. For example, lab collection times at a division. For a single entity (division in this case), multiple collection times may exist. Each collection time would be assigned a unique instance.

A parameter is not considered multi-valued if it can apply to several entities, but for each entity only one value of the parameter exists. For example, "maximum days for a lab order" can be set for every location in the hospital. However, since there is only one value for each location, "maximum days for a lab order" is not multi-valued.

When a parameter that is multi-valued is defined, the instance can be defined as any of the following:

- Numeric
- Date/Time
- Pointer
- Set Of Codes
- Free Text
- Yes/No

The validating logic for an instance is defined the same way as for a value.

## Value

A value can be assigned to every parameter for the entities allowed in the parameter definition. Values are stored in the PARAMETERS file (#8989.5). Fields in the PARAMETERS file (#8989.5) map to DIR fields. DIR is used to validate the data. Values can be any of the following:

- Numeric
- Date/Time
- Pointer
- Set Of Codes
- Free Text
- Yes/No
- Word-processing Type

## Parameter Template

A Parameter template is similar to an Input template. It contains a list of parameters that can be entered through an input session (e.g., an option). Templates are stored in the PARAMETER TEMPLATE file (#8989.52). Entries in this file must also be namespaced.

There are two Input templates for adding parameter definitions:

Template	Description
XPAR SINGLE VALUED CREATE	For adding/editing parameters that will be single valued
XPAR MULTI VALUED CREATE	For adding/editing parameters that will be multiple valued

**Table 1-2: Templates—Parameter Tools**

## Why Would You Use Parameter Tools?

The reason a developer would use Parameter Tools is to allow a hierarchical designation of a parameter value. Thus, rather than many parameters that exist now, which are just for the system level or just for a particular clinic, Parameter Tools allows you to define:

- Different levels at which the parameter can be set.
- In what priority the values are used.

Take for example setting up a default order menu for a person. Each facility may have a default order menu for their primary care clinicians. Each division may have one that is slightly different if their

practices vary enough. For each location, they may set up a different order menu so that users working in a cardiology clinic get a different set of possible orders than those in a dermatology clinic. And there may be reasons to give one specific person a different order menu because they are authorized to prescribe additional medications, because they tend to practice in a different flow, or for other reasons. It's one parameter, but it allows the parameter to be set for multiple entities (at multiple levels). Those entities are defined in the IA, but can include package (PKG, which only developers should set—these are default export values), system (SYS, whole medical facility), division (DIV), location (LOC), room-bed (BED), team (TEA), provider, etc.

The PARAMETER DEFINITION file (#8989.51) defines what entities are allowed to be used for a parameter and in which order they are resolved (individual takes precedence over location takes precedence over division takes precedence over system which takes precedence over package). Sometimes you would want to create defaults for your medical center, but allow users in a certain area to customize what they see and do for their particular role.

XPAR finds the appropriate value based on the parameter definitions and settings that may exist. This way, the developer does not need to look at multiple different location or person files to determine how the software should operate.

With integrations, this is even more important because it allows facilities to integrate but, at the same time, continue some business practices based on parameters set at the division level rather than at the system level.

## Example

The following is a simple example of a way you might use the Parameter Tools.

Suppose you needed a parameter that could be set as a default for the system (account) and also overridden for a given user. Previously, you had to add a field to a software site file (e.g., the KERNEL SYSTEM PARAMETERS file [#8989.3]) and then add a similar field to the NEW PERSON file (#200). This situation is a perfect use of the Parameter Tools.

1. You need the equivalent to a data dictionary (DD) entry. This goes into the PARAMETER DEFINITION file (#8989.51). In this case we need a Yes/No Set of Codes. So this is what you set up:

```
Name: XUS-XUP VPE
DISPLAY TEXT: Drop into VPE
MULTIPLE VALUED: No
VALUE DATA TYPE: yes/no
VALUE HELP: Should XUP drop the user into the VPE environment?
Description...
PRECEDENCE: 1      ENTITY FILE: USER
PRECEDENCE: 2      ENTITY FILE: SYSTEM
```

This last item gives the order that values are looked for and/or returned. You want a USER value (File #200) if there is one; otherwise a SYSTEM value (File #4.2). It also gives the entities that are allowed to have values of this data. In the place of SYSTEM, you could have used PACKAGE.

2. Now you can use ^XPAREDIT to enter a value for your new parameter:

```
>D ^XPAREDIT

      --- Edit Parameter Values ---

Select PARAMETER DEFINITION NAME: XUS-XUP VPE      Drop into VPE

XUS-XUP VPE may be set for the following:

 1   User          USR      [choose from NEW PERSON]
 2   System        SYS      [NXT.KERNEL.ISC-SF.VA.GOV]

Enter selection: 2 <Enter>  System  NXT.KERNEL.ISC-SF.VA.GOV

----- Setting XUS-XUP VPE  for System: NXT.KERNEL.ISC-SF.VA.GOV -----
Value:  NO
...
```

3. Now how do you get this value out in your program?

```
>S X=$$GET^XPAR("USR^SYS","XUS-XUP VPE",1,"Q") ;X will be null, 0 or 1.
```

For the first parameter, you want a value from USR (user / New Person) or SYS (system)  
Next, this is the name of the parameter: "**XUS-XUP VPE**"

Next, in this example you only allow one instance (optional, Defaults to 1 if not passed in).

Last, the format to return: you use "Q" to get, in the quickest manner, the internal value.

Adding the parameter template with VA FileMan:

```
Select PARAMETER DEFINITION NAME: XUS-XUP VPE Drop into VPE

NAME: XUS-XUP VPE// <Enter>
DISPLAY TEXT: Drop into VPE// <Enter>
MULTIPLE VALUED: No// <Enter>
INSTANCE TERM: <Enter>
VALUE TERM: <Enter>
PROHIBIT EDITING: <Enter>
VALUE DATA TYPE: yes/no// <Enter>
VALUE DOMAIN: <Enter>
VALUE HELP: Should XUP drop the user into the VPE environment.
VALUE VALIDATION CODE: <Enter>
VALUE SCREEN CODE: <Enter>
INSTANCE DATA TYPE: <Enter>
INSTANCE DOMAIN: <Enter>
INSTANCE HELP: <Enter>
INSTANCE VALIDATION CODE: <Enter>
INSTANCE SCREEN CODE: <Enter>
DESCRIPTION:
    1> This parameter controls if a user when exiting XUP is dropped into
    2> VPE or right to the ">" prompt.
EDIT Option: <Enter>
Select PRECEDENCE: 2// <Enter>
PRECEDENCE: 2// <Enter>
ENTITY FILE: SYSTEM// <Enter>
Select PRECEDENCE: <Enter>
```

**Figure 1-1: Adding a parameter template (sample)**

# Chapter 2: Programmer Manual Information

This is the Programmer Manual section of this supplemental documentation for the Parameter Tools software (i.e., Kernel Toolkit Patch XT\*7.3\*26). It will be incorporated into the *Kernel Toolkit User Manual* at a later date.

The intended audience for this chapter is the application developers of VistA software. However, it can also be helpful to others in Information Resource Management (IRM), Enterprise VistA Support (EVS), and Application Structure and Integration Services (ASIS).

## **Application Program Interfaces (APIs)—^XPAR Routine**

The following is a list of APIs available in the ^XPAR routine.

### **\$\$GET^XPAR—Return An Instance of a Parameter**

**Category:** Parameter Tools

**Reference Type:** Supported

**Integration Agreement Number:** 2263

**Description:**

This extrinsic function allows you to retrieve the value of a parameter. The value is returned from this API in the format defined by the "format" input parameter (see below).

**Format:**

```
$$GET^XPAR(entity,parameter[,instance][,format])
```

**Input Parameters:**

<b>entity:</b>	(required) Entity is defined as the single entity or group of entities you want to look at in order to retrieve the value. Entities may be passed in internal or external format (e.g. LOC.PULMONARY or LOC.'57 or 57;SC()). The list of entities in this variable may be defined as follows: <ol style="list-style-type: none"> <li>1. A single entity to look at (e.g. LOC.PULMONARY).</li> <li>2. The word "ALL" that will tell the utility to look for values assigned to the parameter using the entity precedence defined in the PARAMETER DEFINITION file (#8989.51).</li> <li>3. A list of entities you want to search (e.g. "USR^LOC^SYS^PKG"). The list is searched from left to right with the first value found returned.</li> <li>4. Items 2 or 3 with specific entity values referenced such as:               <ul style="list-style-type: none"> <li>• ALL^LOC.PULMONARY—to look at the defined entity precedence, but when looking at location, only look at the PULMONARY location.</li> <li>• USR^LOC.PULMONARY^SYS^PKG—to look for values for all current user, PULMONARY location, system, or package).</li> </ul> </li> </ol>
<b>parameter:</b>	(required) Parameter identifies the name or Internal Entry Number (IEN) of the parameter as defined in the PARAMETER DEFINITION file (#8989.51).
<b>instance:</b>	(optional) Defaults to 1 if not passed in. Can be passed in external or internal format. Internal format requires that the value be preceded by the "" character).
<b>format:</b>	(optional) Defaults to "Q" if not defined. Format determines how the value is returned. It can be set to the following: "I"—Internal, returns internal value. "Q"—Quick, returns the value in the quickest manner—internal format. "E"—External, returns external value. "B"—Both, returns internal^external value.

**Output:**

<b>returns:</b>	Returns parameter value in pre-defined format.
-----------------	--

**Example:**

Example forthcoming...

## **ADD^XPAR—Add Parameter Value**

**Category:** Parameter Tools

**Reference Type:** Supported

**Integration Agreement Number:** 2263

**Description:**

This M-based API is called to add a new parameter value.

**Format:**

```
ADD^XPAR(entity,parameter[,instance],value[,.error])
```

**Input Parameters:**

<b>entity:</b>	<p>(required) The entity may be set to any of the following:</p> <ul style="list-style-type: none"> <li>• Internal variable pointer (nnn;GLO(123,)).</li> <li>• External format of the variable pointer using the 3-character prefix (prefix.entryname).</li> <li>• Prefix alone to set the parameter based on current entity selected.</li> </ul> <p>This works for the following entities:</p> <p style="padding-left: 40px;">"USR"—Uses current value of DUZ.</p> <p style="padding-left: 40px;">"DIV"—Uses current value of DUZ(2).</p> <p style="padding-left: 40px;">"SYS"—Uses system (domain).</p> <p style="padding-left: 40px;">"PKG"—Uses the package (software) to which the parameter belongs.</p> <p>Entity can also be a list of entities delimited by "^" or the word "ALL". The list of entities is processed in left to right order and the first existing instance is returned. The word "ALL" uses the entity precedence defined by the PARAMETER DEFINITION file (#8989.51) and looks for values in that order. This is useful when the default values are used.</p> <p>For example, to search for a parameter instance, using the precedence defined in the PARAMETER DEFINITION file (#8989.51) and taking default entity values:</p> <pre><b>MYENT="ALL"</b></pre> <p>To do the same as above, but explicitly name a value for one of the entities:</p> <pre><b>MYENT="ALL^LOC.PULMONARY CLINIC"</b></pre> <p>To explicitly name the search order and entity values:</p> <pre><b>MYENT="USR^LOC.PULMONARY CLINIC^SYS^DIV^PKG"</b></pre> <p>Or using internal values, variable pointers, etc.:</p> <pre><b>MYENT="USR.`1234^LOC.`57^SYS^34;DIC(4,^PKG"</b></pre> <p>Of course, you may also just pass a single entity in external or internal variable pointer format:</p> <pre><b>MYENT="LOC.PULMONARY CLINIC"</b></pre>
<b>parameter:</b>	<p>(required) Identifies the name or Internal Entry Number (IEN) of the parameter as defined in the PARAMETER DEFINITION file (#8989.51).</p>

<b>instance:</b>	(optional) Defaults to 1. Can be passed in external or internal format. Internal format requires that the value be preceded by the `'' character.
<b>value:</b>	(required) Can be passed in external or internal format. If using internal format for a pointer type parameter, the value must be preceded with the `` character.).  If the value is being assigned to a word-processing parameter, the text can be passed in the subordinate nodes of Value (e.g. Value(1,0)=Text) and the variable "Value" itself can be defined as a title or description of the text.

**Output:**

<b>.error:</b>	(optional) If used, it must be passed in by reference. It returns any error condition that may occur. If no error occurs, the value assigned will be 0 (zero). If an error does occur, it will be in the format: "#^errortext".  The # is the number in the VA FileMan DIALOG file (#.84) and the errortext describes the error.
----------------	--

**Example:**

Example forthcoming...

**CHG^XPAR—Change Parameter Value**

**Category:** Parameter Tools

**Reference Type:** Supported

**Integration Agreement Number:** 2263

**Description:**

This M-based API is called to change an existing parameter value.

**Format:**

```
CHG^XPAR(entity,parameter[,instance],value[,.error])
```

**Input Parameters:**

<b>entity:</b>	<p>(required) The entity may be set to any of the following:</p> <ul style="list-style-type: none"> <li>• Internal variable pointer (nnn;GLO(123,)).</li> <li>• External format of the variable pointer using the 3-character prefix (prefix.entryname).</li> <li>• Prefix alone to set the parameter based on current entity selected.</li> </ul> <p>This works for the following entities:</p> <p style="padding-left: 40px;">"USR"—Uses current value of Duz.</p> <p style="padding-left: 40px;">"DIV"—Uses current value of Duz(2).</p> <p style="padding-left: 40px;">"SYS"—Uses system (domain).</p> <p style="padding-left: 40px;">"PKG"—Uses the package (software) to which the parameter belongs.</p> <p>Entity can also be a list of entities delimited by "^" or the word "ALL". The list of entities is processed in left to right order and the first existing instance is returned. The word "ALL" uses the entity precedence defined by the PARAMETER DEFINITION file (#8989.51) and looks for values in that order. This is useful when the default values are used.</p> <p>For example, to search for a parameter instance, using the precedence defined in the PARAMETER DEFINITION file (#8989.51) and taking default entity values:</p> <pre><b>MYENT="ALL"</b></pre> <p>To do the same as above, but explicitly name a value for one of the entities:</p> <pre><b>MYENT="ALL^LOC.PULMONARY CLINIC"</b></pre> <p>To explicitly name the search order and entity values:</p> <pre><b>MYENT="USR^LOC.PULMONARY CLINIC^SYS^DIV^PKG"</b></pre> <p>Or using internal values, variable pointers, etc.:</p> <pre><b>MYENT="USR.`1234^LOC.`57^SYS^34;DIC(4,^PKG"</b></pre> <p>Of course, you may also just pass a single entity in external or internal variable pointer format:</p> <pre><b>MYENT="LOC.PULMONARY CLINIC"</b></pre>
<b>parameter:</b>	<p>(required) Identifies the name or Internal Entry Number (IEN) of the parameter as defined in the PARAMETER DEFINITION file (#8989.51).</p>

<b>instance:</b>	(optional) Defaults to 1. Can be passed in external or internal format. Internal format requires that the value be preceded by the `'' character.
<b>value:</b>	(required) Can be passed in external or internal format. If using internal format for a pointer type parameter, the value must be preceded with the `'' character.).  If the value is being assigned to a word-processing parameter, the text can be passed in the subordinate nodes of Value (e.g. Value(1,0)=Text) and the variable "Value" itself can be defined as a title or description of the text.

**Output:**

<b>.error:</b>	(optional) If used, it must be passed in by reference. It returns any error condition that may occur. If no error occurs, the value assigned will be 0 (zero). If an error does occur, it will be in the format: "#^errortext".  The # is the number in the VA FileMan DIALOG file (#.84) and the errortext describes the error.
----------------	--

**Example:**

Example forthcoming...

**DEL^XPAR—Delete Parameter Value**

**Category:** Parameter Tools

**Reference Type:** Supported

**Integration Agreement Number:** 2263

**Description:**

This M-based API is called to delete a parameter value.

**Format:**

```
DEL^XPAR(entity,parameter[,instance][,.error])
```

**Input Parameters:**

<b>entity:</b>	<p>(required) The entity may be set to any of the following:</p> <ul style="list-style-type: none"> <li>• Internal variable pointer (nnn;GLO(123,)).</li> <li>• External format of the variable pointer using the 3-character prefix (prefix.entryname).</li> <li>• Prefix alone to set the parameter based on current entity selected.</li> </ul> <p>This works for the following entities:</p> <p style="padding-left: 40px;">"USR"—Uses current value of Duz.</p> <p style="padding-left: 40px;">"DIV"—Uses current value of Duz(2).</p> <p style="padding-left: 40px;">"SYS"—Uses system (domain).</p> <p style="padding-left: 40px;">"PKG"—Uses the package (software) to which the parameter belongs.</p> <p>Entity can also be a list of entities delimited by "^" or the word "ALL". The list of entities is processed in left to right order and the first existing instance is returned. The word "ALL" uses the entity precedence defined by the PARAMETER DEFINITION file (#8989.51) and looks for values in that order. This is useful when the default values are used.</p> <p>For example, to search for a parameter instance, using the precedence defined in the PARAMETER DEFINITION file (#8989.51) and taking default entity values:</p> <pre><b>MYENT="ALL"</b></pre> <p>To do the same as above, but explicitly name a value for one of the entities:</p> <pre><b>MYENT="ALL^LOC.PULMONARY CLINIC"</b></pre> <p>To explicitly name the search order and entity values:</p> <pre><b>MYENT="USR^LOC.PULMONARY CLINIC^SYS^DIV^PKG"</b></pre> <p>Or using internal values, variable pointers, etc.:</p> <pre><b>MYENT="USR.`1234^LOC.`57^SYS^34;DIC(4,^PKG"</b></pre> <p>Of course, you may also just pass a single entity in external or internal variable pointer format:</p> <pre><b>MYENT="LOC.PULMONARY CLINIC"</b></pre>
<b>parameter:</b>	<p>(required) Identifies the name or Internal Entry Number (IEN) of the parameter as defined in the PARAMETER DEFINITION file (#8989.51).</p>

<b>instance:</b>	(optional) Defaults to 1. Can be passed in external or internal format. Internal format requires that the value be preceded by the "" character.
------------------	---

**Output:**

<b>.error:</b>	(optional) If used, it must be passed in by reference. It returns any error condition that may occur. If no error occurs, the value assigned will be 0 (zero). If an error does occur, it will be in the format: "#^errortext".  The # is the number in the VA FileMan DIALOG file (#.84) and the errortext describes the error.
----------------	--

**Example:**

Example forthcoming...

**EN^XPAR—Add, Change, Delete Parameters**

**Category:** Parameter Tools

**Reference Type:** Supported

**Integration Agreement Number:** 2263

**Description:**

This M-based API does any of the following:

- Add the value as a new entry to the PARAMETERS file (#8989.5) if the Entity|Parameter|Instance combination does not already exist.
- Change the value assigned to the parameter in the PARAMETERS file (#8989.5) if the Entity|Parameter|Instance combination already exists.
- Delete the parameter instance in the PARAMETERS file (#8989.5) if the value assigned is "@".

**Format:**

```
EN^XPAR(entity,parameter[,instance],value[,.error])
```

**Input Parameters:**

<b>entity:</b>	<p>(required) The entity may be set to any of the following:</p> <ul style="list-style-type: none"> <li>• Internal variable pointer (nnn;GLO(123,)).</li> <li>• External format of the variable pointer using the 3-character prefix (prefix.entryname).</li> <li>• Prefix alone to set the parameter based on current entity selected.</li> </ul> <p>This works for the following entities:</p> <p style="padding-left: 40px;">"USR"—Uses current value of Duz.</p> <p style="padding-left: 40px;">"DIV"—Uses current value of Duz(2).</p> <p style="padding-left: 40px;">"SYS"—Uses system (domain).</p> <p style="padding-left: 40px;">"PKG"—Uses the package (software) to which the parameter belongs.</p> <p>Entity can also be a list of entities delimited by "^" or the word "ALL". The list of entities is processed in left to right order and the first existing instance is returned. The word "ALL" uses the entity precedence defined by the PARAMETER DEFINITION file (#8989.51) and looks for values in that order. This is useful when the default values are used.</p> <p>For example, to search for a parameter instance, using the precedence defined in the PARAMETER DEFINITION file (#8989.51) and taking default entity values:</p> <pre><b>MYENT="ALL"</b></pre> <p>To do the same as above, but explicitly name a value for one of the entities:</p> <pre><b>MYENT="ALL^LOC.PULMONARY CLINIC"</b></pre> <p>To explicitly name the search order and entity values:</p> <pre><b>MYENT="USR^LOC.PULMONARY CLINIC^SYS^DIV^PKG"</b></pre> <p>Or using internal values, variable pointers, etc.:</p> <pre><b>MYENT="USR.^1234^LOC.^57^SYS^34;DIC(4,^PKG"</b></pre> <p>Of course, you may also just pass a single entity in external or internal variable pointer format:</p> <pre><b>MYENT="LOC.PULMONARY CLINIC"</b></pre>
<b>parameter:</b>	<p>(required) Identifies the name or Internal Entry Number (IEN) of the parameter as defined in the PARAMETER DEFINITION file (#8989.51).</p>

<b>instance:</b>	(optional) Defaults to 1. Can be passed in external or internal format. Internal format requires that the value be preceded by the `'' character.
<b>value:</b>	(required) Can be passed in external or internal format. If using internal format for a pointer type parameter, the value must be preceded with the `` character.).  If the value is being assigned to a word-processing parameter, the text can be passed in the subordinate nodes of Value (e.g. Value(1,0)=Text) and the variable "Value" itself can be defined as a title or description of the text.

**Output:**

<b>.error:</b>	(optional) If used, it must be passed in by reference. It returns any error condition that may occur. If no error occurs, the value assigned will be 0 (zero). If an error does occur, it will be in the format: "#^errortext".  The # is the number in the VA FileMan DIALOG file (#.84) and the errortext describes the error.
----------------	--

**Example:**

Example forthcoming...

**ENVAL^XPAR—Return All Instances of a Parameter**

**Category:** Parameter Tools

**Reference Type:** Supported

**Integration Agreement Number:** 2263

**Description:**

This M-based API returns all instances of a parameter.

**Format:**

```
ENVAL^XPAR(.list,parameter[,instance][,.error][,gbl])
```

**Input Parameters:**

<b>.list</b>	(required) Name of array to hold the output. When used with the optional "gbl" input parameter it is ".list", without "gbl" it is a closed global root.
<b>parameter:</b>	(required) Parameter identifies the name or Internal Entry Number (IEN) of the parameter as defined in the PARAMETER DEFINITION file (#8989.51).
<b>instance:</b>	(optional) Defaults to 1 if not passed in. Can be passed in external or internal format. Internal format requires that the value be preceded by the "" character).
<b>gbl</b>	(optional) Set to 1 if "list" holds a closed global root.

**Output:**

<b>.list</b>	The array passed as "list" is returned with all of the possible values assigned to the parameter. Data is returned in the list(ent,inst)=val format.
<b>.error:</b>	(optional) If used, it must be passed in by reference. It returns any error condition that may occur. If no error occurs, the value assigned will be 0 (zero). If an error does occur, it will be in the format: "#^errortext". The # is the number in the VA FileMan DIALOG file (#.84) and the errortext describes the error.

**Example:**

Example forthcoming...

## GETLST^XPAR—Return All Instances of a Parameter

**Category:** Parameter Tools

**Reference Type:** Supported

**Integration Agreement Number:** 2263

**Description:**

This M-based API is similar to \$\$GET^XPAR—Return An Instance of a Parameter *except* this API returns *all* instances of a parameter.

**Format:**

```
GETLST^XPAR(.list,ent,par,fmt[,err][,gbl])
```

**Input Parameters:**

<b>.list</b>	(required) Name of array to hold the output. When used with the optional "gbl" input parameter it is ".list", without "gbl" it is a closed global root.
<b>ent:</b>	(required) ENT is defined as the single entity or group of entities you want to look at in order to retrieve the value. Entities may be passed in internal or external format (e.g. LOC.PULMONARY or LOC.'57 or 57;SC()). The list of entities in this variable may be defined as follows: <ol style="list-style-type: none"> <li>1. A single entity to look at (e.g. LOC.PULMONARY).</li> <li>2. The word "ALL" that will tell the utility to look for values assigned to the parameter using the entity precedence defined in the PARAMETER DEFINITION file (#8989.51).</li> <li>3. A list of entities you want to search (e.g. "USR^LOC^SYS^PKG"). The list is searched from left to right with the first value found returned.</li> <li>4. Items 2 or 3 with specific entity values referenced such as: <ul style="list-style-type: none"> <li>• ALL^LOC.PULMONARY—to look at the defined entity precedence, but when looking at location, only look at the PULMONARY location.</li> <li>• USR^LOC.PULMONARY^SYS^PKG—to look for values for all current user, PULMONARY location, system, or package).</li> </ul> </li> </ol>
<b>par:</b>	(required) Parameter identifies the name or Internal Entry Number (IEN) of the parameter as defined in the PARAMETER

	DEFINITION file (#8989.51).
<b>fmt:</b>	<p>(optional) Defaults to "Q" if not defined.</p> <p>FMT determines how the value is returned. It can be set to the following:</p> <ul style="list-style-type: none"> <li>"I"—Internal, returns internal value: list(instance)=internal value.</li> <li>"Q"—Quick, returns the value in the quickest manner: list(#)=internal instance^internal value.</li> <li>"E"—External, returns external value: list(#)=external instance^external value.</li> <li>"B"—Both, returns both internal and external values:           <ul style="list-style-type: none"> <li>• list(#,"N")=internal instance^external instance.</li> <li>• list(#,"V")=internal value^external value.</li> </ul> </li> <li>"N"—External instance: list("N")=internal value^external value.</li> </ul>
<b>gbl:</b>	(optional) Set to 1 if "list" holds a closed global root.

**Output:**

<b>.list:</b>	The array passed as "list" is returned with all of the possible values assigned to the parameter. Data is returned in the list(ent,inst)=val format.
<b>.err:</b>	(optional) If used, it must be passed in by reference. It returns any error condition that may occur. If no error occurs, the value assigned will be 0 (zero). If an error does occur, it will be in the format: "#^errortext".  The # is the number in the VA FileMan DIALOG file (#.84) and the errortext describes the error.

**Example:**

Example forthcoming...

## GETWP^XPAR—Return Word-processing Text

**Category:** Parameter Tools

**Reference Type:** Supported

**Integration Agreement Number:** 2263

**Description:**

This M-based API returns word-processing text in the "returnedtext" parameter. The "returnedtext" parameter itself contains the value field, which is free text that may contain a title, description, or other text. The word-processing text is returned in returnedtext(#,0).



For an example of the word-processing text returned, please refer to the "returnedtext" output below.

**Format:**

```
GETWP^XPAR(returnedtext,entity,parameter[,instance][,.error])
```

**Input Parameters:**

<b>returnedtext:</b>	(required) This parameter is defined as the name of an array in which you want the text returned. It is set to the title, description, etc. The actual word-processing text is returned in returnedtext(#,0). For Example:  returnedtext="Select Notes Help" returnedtext(1,0)="To select a progress note from the list," returnedtext(2,0)="click on the date/title of the note."
<b>entity:</b>	(required) Entity is defined as the single entity or group of entities you want to look at in order to retrieve the value. Entities may be passed in internal or external format (e.g. LOC.PULMONARY or LOC.'57 or 57;SC()). The list of entities in this variable may be defined as follows:  1. A single entity to look at (e.g. LOC.PULMONARY). 2. The word "ALL" that will tell the utility to look for values assigned to the parameter using the entity precedence defined in the PARAMETER DEFINITION file (#8989.51). 3. A list of entities you want to search (e.g. "USR^LOC^SYS^PKG"). The list is searched from left to right with the first value found returned. 4. Items 2 or 3 with specific entity values referenced such as:

	<ul style="list-style-type: none"> <li>• ALL^LOC.PULMONARY—to look at the defined entity precedence, but when looking at location, only look at the PULMONARY location.</li> <li>• USR^LOC.PULMONARY^SYS^PKG—to look for values for all current user, PULMONARY location, system, or package).</li> </ul>
<b>parameter:</b>	(required) Parameter identifies the name or Internal Entry Number (IEN) of the parameter as defined in the PARAMETER DEFINITION file (#8989.51).
<b>instance:</b>	(optional) Defaults to 1 if not passed in. Can be passed in external or internal format. Internal format requires that the value be preceded by the "" character).

**Output:**

<b>.error:</b>	(optional) If used, it must be passed in by reference. It returns any error condition that may occur. If no error occurs, the value assigned will be 0 (zero). If an error does occur, it will be in the format: "#^errortext".  The # is the number in the VA FileMan DIALOG file (#.84) and the errortext describes the error.
<b>returnedtext:</b>	This variable is defined as the name of an array in which you want the text returned. The returnedtext parameter is set to the title, description, etc. The actual word-processing text is returned in returnedtext(#,0).  For example:  <pre>&gt;D GETWP^XPAR(.X,"PKG","ORW HELP","1stNotes",.ERR)</pre> It might return:  X="Select Notes Help" X(1,0)="To select a progress notes from" X(2,0)="the list, click on the date/title" X(3,0)="of the note."

**Example:**

Example forthcoming...

## **NDEL^XPAR—Delete All Instances of a Parameter**

**Category:** Parameter Tools

**Reference Type:** Supported

**Integration Agreement Number:** 2263

**Description:**

This M-based API is called to delete the value for all instances of a parameter for a given entity.

**Format:**

```
NDEL^XPAR(entity,parameter[, .error])
```

**Input Parameters:**

<b>entity:</b>	<p>(required) The entity may be set to any of the following:</p> <ul style="list-style-type: none"> <li>• Internal variable pointer (nnn;GLO(123,)).</li> <li>• External format of the variable pointer using the 3-character prefix (prefix.entryname).</li> <li>• Prefix alone to set the parameter based on current entity selected.</li> </ul> <p>This works for the following entities:</p> <p style="padding-left: 40px;">"USR"—Uses current value of Duz.</p> <p style="padding-left: 40px;">"DIV"—Uses current value of Duz(2).</p> <p style="padding-left: 40px;">"SYS"—Uses system (domain).</p> <p style="padding-left: 40px;">"PKG"—Uses the package (software) to which the parameter belongs.</p> <p>Entity can also be a list of entities delimited by "^" or the word "ALL". The list of entities is processed in left to right order and the first existing instance is returned. The word "ALL" uses the entity precedence defined by the PARAMETER DEFINITION file (#8989.51) and looks for values in that order. This is useful when the default values are used.</p> <p>For example, to search for a parameter instance, using the precedence defined in the PARAMETER DEFINITION file (#8989.51) and taking default entity values:</p> <pre>MYENT="ALL"</pre> <p>To do the same as above, but explicitly name a value for one of the entities:</p> <pre>MYENT="ALL^LOC.PULMONARY CLINIC"</pre> <p>To explicitly name the search order and entity values:</p> <pre>MYENT="USR^LOC.PULMONARY CLINIC^SYS^DIV^PKG"</pre> <p>Or using internal values, variable pointers, etc.:</p> <pre>MYENT="USR.^1234^LOC.^57^SYS^34;DIC(4,^PKG"</pre> <p>Of course, you may also just pass a single entity in external or internal variable pointer format:</p> <pre>MYENT="LOC.PULMONARY CLINIC"</pre>
<b>parameter:</b>	(required) Identifies the name or Internal Entry Number (IEN) of the parameter as defined in the PARAMETER DEFINITION file (#8989.51).

**Output:**

<b>.error:</b>	(optional) If used, it must be passed in by reference. It returns any error condition that may occur. If no error occurs, the value assigned will be 0 (zero). If an error does occur, it will be in the format: "#^errortext". The # is the number in the VA FileMan DIALOG file (#.84) and the errortext describes the error.
----------------	--

**Example:**

Example forthcoming...

**PUT^XPAR—Add/Update Parameter Instance**

**Category:** Parameter Tools

**Reference Type:** Supported

**Integration Agreement Number:** 2263

**Description:**

This M-based API is called to add or update a parameter instance and bypass the input transforms.

**Format:**

```
PUT^XPAR(entity,parameter[,instance],value[,.error])
```

**Input Parameters:**

<b>entity:</b>	(required) Entity is defined as the single entity or group of entities you want to look at in order to retrieve the value. Entities may be passed in internal or external format (e.g. LOC.PULMONARY or LOC.'57 or 57;SC()). The list of entities in this variable may be defined as follows: <ol style="list-style-type: none"> <li>1. A single entity to look at (e.g. LOC.PULMONARY).</li> <li>2. The word "ALL" that will tell the utility to look for values assigned to the parameter using the entity precedence defined in the PARAMETER DEFINITION file (#8989.51).</li> <li>3. A list of entities you want to search (e.g. "USR^LOC^SYS^PKG"). The list is searched from left to right with the first value found returned.</li> <li>4. Items 2 or 3 with specific entity values referenced such as:               <ul style="list-style-type: none"> <li>• ALL^LOC.PULMONARY—to look at the defined entity precedence, but when looking at location, only look at the PULMONARY location.</li> <li>• USR^LOC.PULMONARY^SYS^PKG—to look for values for all current user, PULMONARY location, system, or package)</li> </ul> </li> </ol>
<b>parameter:</b>	(required) Identifies the name or Internal Entry Number (IEN) of the parameter as defined in the PARAMETER DEFINITION file (#8989.51).
<b>instance:</b>	(optional) Defaults to 1. Can be passed in external or internal format. Internal format requires that the value be preceded by the "" character.
<b>value:</b>	(required) Can be passed in external or internal format. If using internal format for a pointer type parameter, the value must be preceded with the "" character.).  If the value is being assigned to a word-processing parameter, the text can be passed in the subordinate nodes of Value (e.g. Value(1,0)=Text) and the variable "Value" itself can be defined as a title or description of the text.

**Output:**

<b>.error:</b>	(optional) If used, it must be passed in by reference. It returns any error condition that may occur. If no error occurs, the value assigned will be 0 (zero). If an error does occur, it will be in the format: "#^errortext".  The # is the number in the VA FileMan DIALOG file (#.84) and the errortext describes the error.
----------------	--

**Example:**

Example forthcoming...

**REP^XPAR—Replace Instance Value**

**Category:** Parameter Tools

**Reference Type:** Supported

**Integration Agreement Number:** 2263

**Description:**

This M-based API allows a developer to replace the value of an instance with another value.

**Format:**

```
REP^XPAR(entity,parameter,currentinstance,newinstance[, .error])
```

**Input Parameters:**

<b>entity:</b>	<p>(required) The entity may be set to any of the following:</p> <ul style="list-style-type: none"> <li>• Internal variable pointer (nnn;GLO(123,)).</li> <li>• External format of the variable pointer using the 3-character prefix (prefix.entryname).</li> <li>• Prefix alone to set the parameter based on current entity selected.</li> </ul> <p>This works for the following entities:</p> <p style="padding-left: 40px;">"USR"—Uses current value of Duz.</p> <p style="padding-left: 40px;">"DIV"—Uses current value of Duz(2).</p> <p style="padding-left: 40px;">"SYS"—Uses system (domain).</p> <p style="padding-left: 40px;">"PKG"—Uses the package (software) to which the parameter belongs.</p> <p>Entity can also be a list of entities delimited by "^" or the word "ALL". The list of entities is processed in left to right order and the first existing instance is returned. The word "ALL" uses the entity precedence defined by the PARAMETER DEFINITION file (#8989.51) and looks for values in that order. This is useful when the default values are used.</p> <p>For example, to search for a parameter instance, using the precedence defined in the PARAMETER DEFINITION file (#8989.51) and taking default entity values:</p> <pre><b>MYENT="ALL"</b></pre> <p>To do the same as above, but explicitly name a value for one of the entities:</p> <pre><b>MYENT="ALL^LOC.PULMONARY CLINIC"</b></pre> <p>To explicitly name the search order and entity values:</p> <pre><b>MYENT="USR^LOC.PULMONARY CLINIC^SYS^DIV^PKG"</b></pre> <p>Or using internal values, variable pointers, etc.:</p> <pre><b>MYENT="USR.`1234^LOC.`57^SYS^34;DIC(4,^PKG"</b></pre> <p>Of course, you may also just pass a single entity in external or internal variable pointer format:</p> <pre><b>MYENT="LOC.PULMONARY CLINIC"</b></pre>
<b>parameter:</b>	<p>(required) Identifies the name or Internal Entry Number (IEN) of the parameter as defined in the PARAMETER DEFINITION file (#8989.51).</p>

<b>currentinstance:</b>	(required) The instance for which the value is currently defined.
<b>newinstance:</b>	(required) The instance for which you want to assign the value currently assigned to the currentinstance.

**Output:**

<b>.error:</b>	(optional) If used, it must be passed in by reference. It returns any error condition that may occur. If no error occurs, the value assigned will be 0 (zero). If an error does occur, it will be in the format: "#^errortext".  The # is the number in the VA FileMan DIALOG file (#.84) and the errortext describes the error.
----------------	--

**Example:**

Example forthcoming...

## Application Program Interfaces (APIs)—<sup>^</sup>XPAR EDIT Routine

The following is a list of APIs available in the <sup>^</sup>XPAR EDIT routine. The calls are supported for use with the Parameter Tools and are part of the Parameter Tools component of Kernel Toolkit. These calls contain some additional utilities for editing parameters and are covered by IA #2336. (See IA #2263 for the main XPAR entry points to this module.)

### BLDLST<sup>^</sup>XPAR EDIT—Return All Entities of a Parameter

**Category:** Parameter Tools

**Reference Type:** Supported

**Integration Agreement Number:** 2336

**Description:**

This M-based API returns, in the array "list," all entities allowed for the input Parameter.

**Format:**

```
BLDLST^XPAR EDIT(.list,parameter)
```

**Input Parameters:**

<b>.list:</b>	(required) Name of array to receive output.
<b>parameter:</b>	(required) Internal Entry Number (IEN) of entry in the PARAMETER DEFINITION file (#8989.51).

**Output:**

<b>.list:</b>	The array passed as "list" is returned with all of the possible values assigned to the parameter. Data is returned in the list(ent,inst)=val format.
---------------	---

**Example:**

Example forthcoming...

## **EDIT^XPAREDIT—Edit Instance and Value of a Parameter**

**Category:** Parameter Tools

**Reference Type:** Supported

**Integration Agreement Number:** 2336

**Description:**

This M-based API interactively edits the instance (if multiple instances are allowed) and the value for a parameter associated with a given entity.

**Format:**

```
EDIT^XPAREDIT(entity,parameter)
```

**Input Parameters:**

<b>entity:</b>	(required) Identifies the specific entity for which a parameter may be edited. Entity must be in variable pointer format.
<b>parameter:</b>	(required) Identifies the parameter that should be edited. Parameter should contain two pieces: IEN^DisplayNameOfParameter

**Output:**

<b>.list:</b>	The array passed as "list" is returned with all of the possible values assigned to the parameter.   To see how this data can be returned, please refer to the "format" parameter description.
<b>error:</b>	It returns any error condition that may occur. If no error occurs, the value assigned will be 0 (zero). If an error does occur, it will be in the format: "#^errortext".  The # is the number in the VA FileMan DIALOG file (#.84) and the errortext describes the error.

**Example:**

Example forthcoming...

## **EDITPAR^XPAREDIT—Edit Single Parameter**

**Category:** Parameter Tools

**Reference Type:** Supported

**Integration Agreement Number:** 2336

**Description:**

This M-based API is used to edit a single parameter.

**Format:**

EDITPAR^XPAREDIT (parameter)

**Input Parameters:**

<b>parameter:</b>	(required) Pass as the Internal Entry Number (IEN) or the NAME of the entry in the PARAMETER DEFINITION file (#8989.51) that you want to be edited.
-------------------	---

**Output:**

<b>returns:</b>	Returns requested parameter.
-----------------	------------------------------

**Example:**

Example forthcoming...

## **EN^XPAREDIT—Parameter Edit Prompt**

**Category:** Parameter Tools

**Reference Type:** Supported

**Integration Agreement Number:** 2336

**Description:**

This M-based API is called to prompt the user for a parameter to edit. This is provided as a tool for developers and *is not intended for exported calls* as it allows editing of *any* parameter.

**Format:**

```
EN^XPAREDIT
```

**Input Parameters:**

<b>none</b>	
-------------	--

**Output:**

<b>none</b>	
-------------	--

**Example:**

Example forthcoming...

**GETENT^XPAREDIT—Prompt for Entity Based on Parameter**

**Category:** Parameter Tools

**Reference Type:** Supported

**Integration Agreement Number:** 2336

**Description:**

This M-based API interactively prompts for an entity, based on the definition of a parameter.

**Format:**

```
GETENT^XPAREDIT(.entity,parameter,.onlyone?)
```

**Input Parameters:**

<b>parameter:</b>	(required) Specifies the parameter for which an entity should be selected. Parameter should contain two pieces: IEN^DisplayNameOfParameter
-------------------	---

**Output:**

<b>.entity:</b>	(required) Returns the selected entity in variable pointer format.
<b>onlyone?:</b>	<p>(optional) Returns "1" if there is only one possible entity for the value.</p> <p>For example:</p> <ul style="list-style-type: none"> <li>• If the parameter can only be set for the system, onlyone?=1.</li> <li>• If the parameter could be set for any location, onlyone?=0.</li> </ul>

**Example:**

Example forthcoming...

**GETPAR^XPAREDIT—Select Parameter Definition File**

**Category:** Parameter Tools

**Reference Type:** Supported

**Integration Agreement Number:** 2336

**Description:**

This M-based API allows the user to select the PARAMETER DEFINITION file (#8989.51) entry.

**Format:**

```
GETPAR^XPAREDIT(.variable)
```

**Input Parameters:**

<b>.variable</b>	(required) The name of the variable where data is returned.
------------------	---

**Output:**

<b>.variable</b>	Returns the value Y in standard DIC lookup format.
------------------	--

**Example:**

Example forthcoming...

**TED^XPAREDIT—Edit Template Parameters (No Dash Dividers)**

**Category:** Parameter Tools

**Reference Type:** Supported

**Integration Agreement Number:** 2336

**Description:**

This M-based API allows editing of parameters defined in a template. The parameters in the template are prompted in VA FileMan style—prompt by prompt. No dashed line dividers are displayed between each parameter.

Since the dashed line headers are suppressed, it is important to define the VALUE TERM for each parameter in the template, as this is what is used to prompt for the value.

**Format:**

```
TED^XPAREDIT(template[,reviewflags][,allentities])
```

**Input Parameters:**

<b>template:</b>	(required) The Internal Entry Number (IEN) or NAME of an entry in the PARAMETER TEMPLATE file (#8989.52).
<b>reviewflags:</b>	(optional) There are two flags (A and B) that can be used individually, together, or not at all: <ul style="list-style-type: none"> <li>• A—Indicates that the new values for the parameters in the template are displayed <i>after</i> the prompting is done.</li> <li>• B—Indicates that the current values of the parameters are displayed <i>before</i> editing.</li> </ul>
<b>allentities:</b>	(optional) This is a variable pointer that should be used as the entity for all parameters in the template. If left blank, prompting for the entity is done as defined in the PARAMETER TEMPLATE file (#8989.52).

**Output:**

<b>none</b>	
-------------	--

**Example:**

Example forthcoming...

## **TEDH^XPAREDIT—Edit Template Parameters (With Dash Dividers)**

**Category:** Parameter Tools

**Reference Type:** Supported

**Integration Agreement Number:** 2336

**Description:**

This M-based API is similar to the GETENT^XPAREDIT—Prompt for Entity Based on Parameter API except that the dashed line headers *are* shown between each parameter.

It allows editing of parameters defined in a template. The parameters in the template are prompted in VA FileMan style—prompt by prompt.

**Format:**

```
TEDH^XPAREDIT(template[,reviewflags][,allentities])
```

**Input Parameters:**

<b>template:</b>	(required) The Internal Entry Number (IEN) or NAME of an entry in the PARAMETER TEMPLATE file (#8989.52).
<b>reviewflags:</b>	(optional) There are two flags (A and B) that can be used individually, together, or not at all: <ul style="list-style-type: none"><li>• A—Indicates that the new values for the parameters in the template are displayed <i>after</i> the prompting is done.</li><li>• B—Indicates that the current values of the parameters are displayed <i>before</i> editing.</li></ul>
<b>allentities:</b>	(optional) This is a variable pointer that should be used as the entity for all parameters in the template. If left blank, prompting for the entity is done as defined in the PARAMETER TEMPLATE file (#8989.52).

**Output:**

<b>none</b>	
-------------	--

**Example:**

Example forthcoming...



# Index

## A

Add Parameter Value  
ADD^XPAR, 2-3  
Add, Change, Delete Parameters  
EN^XPAR, 2-9  
Add/Update Parameter Instance  
PUT^XPAR, 2-19  
Adobe  
Home Page Web Address, xi  
Adobe Acrobat Quick Guide  
Home Page Web Address, xii  
APIs  
\$\$GET^XPAR (Return An Instance of a Parameter), 2-1  
ADD^XPAR (Add Parameter Value), 2-3  
BLDLST^XPAREDIT (Return All Entities of a Parameter), 2-24  
CHG^XPAR (Change Parameter Value), 2-5  
DEL^XPAR (Delete Parameter Value), 2-7  
EDIT^XPAREDIT (Edit Instance and Value of a Parameter), 2-25  
EDITPAR^XPAREDIT (Edit Single Parameter), 2-26  
EN^XPAR (Add, Change, Delete Parameters), 2-9  
EN^XPAREDIT (Parameter Edit Prompt), 2-26  
ENVAL^XPAR (Return All Instances of a Parameter), 2-11  
GETENT^XPAREDIT (Prompt for Entity Based on Parameter), 2-27  
GETLST^XPAR (Return All Instances of a Parameter), 2-13  
GETPAR^XPAREDIT (Select Parameter Definition File), 2-28  
GETWP^XPAR (Return Word-processing Text), 2-15  
NDEL^XPAR (Delete All Instances of a Parameter), 2-17  
PUT^XPAR (Add/Update Parameter Instance), 2-19  
REP^XPAR (Replace Instance Value), 2-21  
TED^XPAREDIT (Edit Template Parameters [No Dash Dividers]), 2-29  
TEDH^XPAREDIT (Edit Template Parameters [With Dash Dividers]), 2-30

Assumptions About the Reader, xi

## B

Background, 1-2

## C

Change Parameter Value  
CHG^XPAR, 2-5  
Contents, v

## D

Definitions, 1-3  
Delete  
All Instances of a Parameter  
NDEL^XPAR, 2-17  
Parameter Value  
DEL^XPAR, 2-7  
Description, 1-3  
DIALOG File (#.84), 2-5, 2-7, 2-9, 2-11, 2-12, 2-14, 2-16, 2-19, 2-21, 2-23, 2-25  
Documentation  
Revisions, iii

## E

Edit  
Instance and Value of a Parameter  
EDIT^XPAREDIT, 2-25  
Single Parameter  
EDITPAR^XPAREDIT, 2-26  
Template Parameters  
No Dash Dividers  
TED^XPAREDIT, 2-29  
With Dash Dividers  
TEDH^XPAREDIT, 2-30  
Entity  
Definition, 1-3  
Example, 1-6

## F

Figures and Tables, vii  
Files  
DIALOG (#.84), 2-5, 2-7, 2-9, 2-11, 2-12, 2-14, 2-16, 2-19, 2-21, 2-23, 2-25  
KERNEL SYSTEM PARAMETERS (#8989.3), 1-6

- NEW PERSON (#200), 1-2, 1-3, 1-6  
PARAMETER DEFINITION (#8989.51), 1-2, 1-4, 1-6, 2-2, 2-4, 2-6, 2-8, 2-10, 2-12, 2-13, 2-14, 2-15, 2-16, 2-18, 2-20, 2-22, 2-24, 2-26, 2-28  
PARAMETER ENTITY (#8989.518), 1-3  
PARAMETER TEMPLATE (#8989.52), 1-5, 2-29, 2-30  
PARAMETERS (#8989.5), 1-2, 1-4, 1-5, 2-9
- H**
- Help at Prompts, x  
Home Page  
    Adobe Acrobat Quick Guide Home Page Web Address, xii  
    Adobe Home Page Web Address, xi  
    HSD&D Home Page Web Address, xi  
    Kernel Toolkit Home Page Web Address, xi  
    VistA Documentation Library (VDL) Home Page Web Address, xi  
How to  
    Obtain Technical Information Online, x  
    Use this Manual, ix
- HSD&D  
    Home Page Web Address, xi
- I**
- Instance  
    Definition, 1-4  
Introduction, 1-1
- K**
- KERNEL SYSTEM PARAMETERS file (#8989.3), 1-6  
Kernel Toolkit  
    Home Page Web Address, xi
- N**
- NEW PERSON File (#200), 1-2, 1-3, 1-6
- O**
- Obtaining  
    Data Dictionary Listings, x  
    Technical Information Online, How to, x  
Orientation, ix
- P**
- Parameter, 1-5
- Definition, 1-4  
PARAMETER DEFINITION File (#8989.51), 1-2, 1-4, 1-6, 2-2, 2-4, 2-6, 2-8, 2-10, 2-12, 2-13, 2-14, 2-15, 2-16, 2-18, 2-20, 2-22, 2-24, 2-26, 2-28  
Parameter Edit Prompt  
    EN^XPAREDIT, 2-26  
PARAMETER ENTITY File (#8989.518), 1-3  
PARAMETER TEMPLATE File (#8989.52), 1-5, 2-29, 2-30  
Parameter Tools  
APIs  
    \$\$GET^XPAR (Return An Instance of a Parameter), 2-1  
    ADD^XPAR (Add Parameter Value), 2-3  
    BLDLST^XPAREDIT (Return All Entities of a Parameter), 2-24  
    CHG^XPAR (Change Parameter Value), 2-5  
    EDIT^XPAREDIT (Edit Instance and Value of a Parameter), 2-25  
    EDTPAR^XPAREDIT (Edit Single Parameter), 2-26  
    EN^XPAREDIT (Parameter Edit Prompt), 2-26  
    ENVAL^XPAR (Return All Instances of a Parameter), 2-11  
    FEL^XPAR (Delete Parameter Value), 2-7  
    GETENT^XPAREDIT (Prompt for Entity Based on Parameter), 2-27  
    GETLST^XPAR (Return All Instances of a Parameter), 2-13  
    GETPAR^XPAREDIT (Select Parameter Definition File), 2-28  
    GETWP^XPAR (Return Word-processing Text), 2-15  
    NDEL^XPAR (Delete All Instances of a Parameter), 2-17  
    PUT^XPAR (Add/Update Parameter Instance), 2-19  
    REP^XPAR (Replace Instance Value), 2-21  
    TED^XPAREDIT (Edit Template Parameters [No Dash Dividers]), 2-29  
    TEDH^XPAREDIT (Edit Template Parameters [With Dash Dividers]), 2-30  
Background, 1-2  
Definitions, 1-3  
Description, 1-3

Entity Definition, 1-3  
 Example, 1-6  
 Instance Definition, 1-4  
 Introduction, 1-1  
 Parameter Definition, 1-4  
 Template Definition, 1-5  
 Value Definition, 1-5  
 Why Would You Use?, 1-5  
**PARAMETERS** File (#8989.5), 1-2, 1-4, 1-5, 2-9  
 Patches  
   Revisions, iii  
 Programmer Manual Information, 2-1  
 Prompt for Entity Based on Parameter  
   GETENT^XPAREDIT, 2-27

**R**

Reader, Assumptions About the, xi

Reference Materials, xi

Replace Instance Value  
   REP^XPAR, 2-21

Return

- All Entities of a Parameter  
   BLDLST^XPAREDIT, 2-24
- All Instances of a Parameter  
   ENVAL^XPAR, 2-11  
   GETLST^XPAR, 2-13
- An Instance of a Parameter  
   \$\$GET^XPAR, 2-1
- Word-processing Text  
   GETWP^XPAR, 2-15

Revision History, iii

- Documentation, iii
- Patches, iii

Routines

- XPAR  
   APIs, 2-1
- XPAREDIT  
   APIs, 2-24

**S**

Select Parameter Definition File  
   GETPAR^XPAREDIT, 2-28

Supported APIs

- \$\$GET^XPAR (Return An Instance of a Parameter), 2-1
- ADD^XPAR (Add Parameter Value), 2-3
- BLDLST^XPAREDIT (Return All Entities of a Parameter), 2-24
- CHG^XPAR (Change Parameter Value), 2-5

DEL^XPAR (Delete Parameter Value), 2-7  
 EDIT^XPAREDIT (Edit Instance and Value of a Parameter), 2-25  
 EDITPAR^XPAREDIT (Edit Single Parameter), 2-26  
 EN^XPAR (Add, Change, Delete Parameters), 2-9  
 EN^XPAREDIT (Parameter Edit Prompt), 2-26  
 ENVAL^XPAR (Return All Instances of a Parameter), 2-11  
 GETENT^XPAREDIT (Prompt for Entity Based on Parameter), 2-27  
 GETLST^XPAR (Return All Instances of a Parameter), 2-13  
 GETPAR^XPAREDIT (Select Parameter Definition File), 2-28  
 GETWP^XPAR (Return Word-processing Text), 2-15  
 NDEL^XPAR (Delete All Instances of a Parameter), 2-17  
 PUT^XPAR (Add/Update Parameter Instance), 2-19  
 REP^XPAR (Replace Instance Value), 2-21  
 TED^XPAREDIT (Edit Template Parameters [No Dash Dividers]), 2-29  
 TEDH^XPAREDIT (Edit Template Parameters [With Dash Dividers]), 2-30

**T**

Table of Contents, v

Tables and Figures, vii

Templates  
   Definition, 1-5

Toolkit

- APIs for Parameter Tools  
   \$\$GET^XPAR (Return An Instance of a Parameter), 2-1
- ADD^XPAR (Add Parameter Value), 2-3
- BLDLST^XPAREDIT (Return All Entities of a Parameter), 2-24
- CHG^XPAR (Change Parameter Value), 2-5
- DEL^XPAR (Delete Parameter Value), 2-7
- EDIT^XPAREDIT (Edit Instance and Value of a Parameter), 2-25
- EDITPAR^XPAREDIT (Edit Single Parameter), 2-26
- EN^XPAR (Add, Change, Delete Parameters), 2-9

- EN^XPAREDIT (Parameter Edit Prompt), 2-26  
ENVAL^XPAR (Return All Instances of a Parameter), 2-11  
GETENT^XPAREDIT (Prompt for Entity Based on Parameter), 2-27  
GETLST^XPAR (Return All Instances of a Parameter), 2-13  
GETPAR^XPAREDIT (Select Parameter Definition File), 2-28  
GETWP^XPAR (Return Word-processing Text), 2-15  
NDEL^XPAR (Delete All Instances of a Parameter), 2-17  
PUT^XPAR (Add/Update Parameter Instance), 2-19  
REP^XPAR (Replace Instance Value), 2-21  
TED^XPAREDIT (Edit Template Parameters [No Dash Dividers]), 2-29  
TEDH^XPAREDIT (Edit Template Parameters [With Dash Dividers]), 2-30
- U**  
User Manual Information, 1-1
- V**  
Value  
    Definition, 1-5  
VistA Documentation Library (VDL)  
    Home Page Web Address, xi
- W**  
Web Page  
    Adobe Acrobat Quick Guide Home Page Web Address, xii  
    Adobe Home Page Web Address, xi  
    HSD&D Home Page Web Address, xi  
    Kernel Toolkit Home Page Web Address, xi  
    VistA Documentation Library (VDL) Home Page Web Address, xi  
Why Would You Use Parameter Tools?, 1-5
- X**  
XPAR  
    APIs  
        \$\$GET^XPAR (Return An Instance of a Parameter), 2-1  
        ADD^XPAR (Add Parameter Value), 2-3  
        CHG^XPAR (Change Parameter Value), 2-5  
        DEL^XPAR (Delete Parameter Value), 2-7  
        EN^XPAR (Add, Change, Delete Parameters), 2-9  
        ENVAL^XPAR (Return All Instances of a Parameter), 2-11  
        GETLST^XPAR (Return All Instances of a Parameter), 2-13  
        GETWP^XPAR (Return Word-processing Text), 2-15  
        NDEL^XPAR (Delete All Instances of a Parameter), 2-17  
        PUT^XPAR (Add/Update Parameter Instance), 2-19  
        REP^XPAR (Replace Instance Value), 2-21  
    Routine  
        APIs, 2-1
- XPAREDIT  
    APIs  
        BLDLST^XPAREDIT (Return All Entities of a Parameter), 2-24  
        EDIT^XPAREDIT (Edit Instance and Value of a Parameter), 2-25  
        EDITPAR^XPAREDIT (Edit Single Parameter), 2-26  
        EN^XPAREDIT (Parameter Edit Prompt), 2-26  
        GETENT^XPAREDIT (Prompt for Entity Based on Parameter), 2-27  
        GETPAR^XPAREDIT (Select Parameter Definition File), 2-28  
        TED^XPAREDIT (Edit Template Parameters [No Dash Dividers]), 2-29  
        TEDH^XPAREDIT (Edit Template Parameters [With Dash Dividers]), 2-30  
    Routine  
        APIs, 2-24